**ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ**
**ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**

# ΔΟΜΕΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΑΡΧΕΙΩΝ

## 1η άσκηση

**Ημερομηνία παράδοσης:** 28 Μαρτίου 2021
**Η άσκηση είναι ατομική**

## Επεξεργασία Αρχείων

Σκοπός της άσκησης είναι η εξοικείωση με χρήση αρχείων δίσκο. Η πληροφορία οργανώνεται σε σελίδες δίσκου. Για τους σκοπούς της άσκησης το μέγεθος της σελίδας είναι **page_size=128 bytes**.

## Κλάση Αρχείο (2 μονάδες)

Κατασκευάστε την κλάση «**FileManager**» με λειτουργίες:

- **FileHandle**: αναπαριστάνει ένα ανοικτό αρχείο, την τρέχουσα θέση δείκτη αρχείου, αριθμό σελίδων στο αρχείο, είδος πληροφορίας που αποθηκεύεται και οποιαδήποτε άλλη χρήσιμη πληροφορία που θα χρειαστεί μια εφαρμογή που θα χρησιμοποιήσει το αρχείο. Αυτή η πληροφορία μπορεί γράφεται στην πρώτη σελίδα του αρχείου (ώστε να είναι διαθέσιμη μόλις διαβάσετε την πρώτη σελίδα).
- **CreateFile**: αρχικοποίηση αρχείου, δηλαδή δημιουργεί ένα αρχείο χωρίς δεδομένα με μια σελίδα στην αρχή που αποθηκεύει την πληροφορία που υπάρχει στο FileHandle. Το όνομα του αρχείου στο δίσκο δίνεται ως όρισμα. Ο αριθμός σελίδων παίρνει τιμή 0.
- **OpenFile**: Άνοιγμα αρχείου για ανάγνωση ή εγγραφή, επιστρέφει τον αριθμό σελίδων του αρχείου. Το όνομα του αρχείου στο δίσκο δίνεται ως όρισμα.
- **ReadBlock**: Ανάγνωση σελίδας στο buffer (στην κεντρική μνήμη) στη θέση αρχείου που δίνεται ως παράμετρος.
- **ReadNextBlock**: Ανάγνωση επόμενης σελίδας (ως προς την τρέχουσα θέση) στο buffer
- **ReadPrevBlock**: Ανάγνωση προηγούμενης σελίδας (ως προς την τρέχουσα θέση) στο buffer. Δεν θα την χρειαστείτε και μην την υλοποιήσετε.
- **WriteBlock**: Εγγραφή του buffer στην θέση αρχείου που δίνεται ως παράμετρος. Δεν αυξάνεται ο αριθμός σελίδων του αρχείου αν η εγγραφή γίνεται πάνω σε σελίδα που υπάρχει ήδη.
- **WriteNextBlock**: Εγγραφή buffer στην επόμενη σελίδα αρχείου (ως προς την τρέχουσα θέση). Δεν αυξάνεται ο αριθμός σελίδων του αρχείου αν η εγγραφή γίνεται πάνω σε σελίδα που υπάρχει ήδη.
- **AppendBlock**: Εγγραφή buffer στο τέλος του αρχείου. Αυξάνεται ο αριθμός σελίδων του αρχείου. Είναι ειδική περίπτωση την προηγούμενης.
- **DeleteBlock**: Διαγραφή μίας σελίδας από το αρχείο. Η διαγραφή γίνεται αντιγράφοντας την τελευταία σελίδα του αρχείου στην θέση της σελίδας που διαγράφεται και μειώνοντας κατά ένα τον αριθμό σελίδων του αρχείου. Πρέπει να ενημερώνεται και η FileHandle για τον νέο αριθμό σελίδων.
- **CloseFile**: Ενημέρωση και κλείσιμο του αρχείου σε ένα νέο με τα δεδομένα που έχει εκείνη την στιγμή το FileHandle.

Κάθε λειτουργία επιστρέφει 0 αν για κάποιο λόγο αποτύχει και 1 αν επιτύχει.

**Οργάνωση πληροφορίας σε σειριακό αρχείο και απόδοση αναζήτησης (3 μονάδες)**

**Α τρόπος οργάνωσης αρχείου:** Κάθε σελίδα του αρχείου περιέχει έναν αριθμό από εγγραφές (records). Κάθε εγγραφή έχει μέγεθος **rec_size = 32 bytes**. Επομένως, κάθε σελίδα δίσκου χωράει 4 εγγραφές (4x32=128). Κάθε εγγραφή αποθηκεύει σε δυαδική μορφή (και όχι text) έναν ακέραιο αριθμό «κλειδί» (key) με μέγεθος 4 bytes και κάποια άλλη πληροφορία 28 bytes. Το είδος της πληροφορίας εξαρτάται από την εφαρμογή και μπορεί να είναι οτιδήποτε (π.χ. τυχαίοι αριθμοί ή χαρακτήρες). Στο σχήμα φαίνεται ένα αρχείο με 4 σελίδες. Κάθε σελίδα **αρχείου περιέχει 4 εγγραφές** (τα κλειδιά είναι στην πρώτη γραμμή σε τυχαία σειρά και η πληροφορία παριστάνεται με ένα γράμμα που αντιστοιχεί σε 28 bytes). Ένα αρχείο μπορεί να περιέχει έναν πολύ μεγάλο αριθμό από τέτοιες εγγραφές που αποθηκεύονται σε διαδοχικές σελίδες δίσκου (η μία μετά την άλλη).

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 13 | 3 | 2 | 50 | 60 | 35 | 20 | 4 | 6 | 21 | 33 | 8 | 22 | 9 | 11 |
| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p |
| Σελίδα 0 | | | | Σελίδα 1 | | | | Σελίδα 2 | | | | Σελίδα 3 | | | |

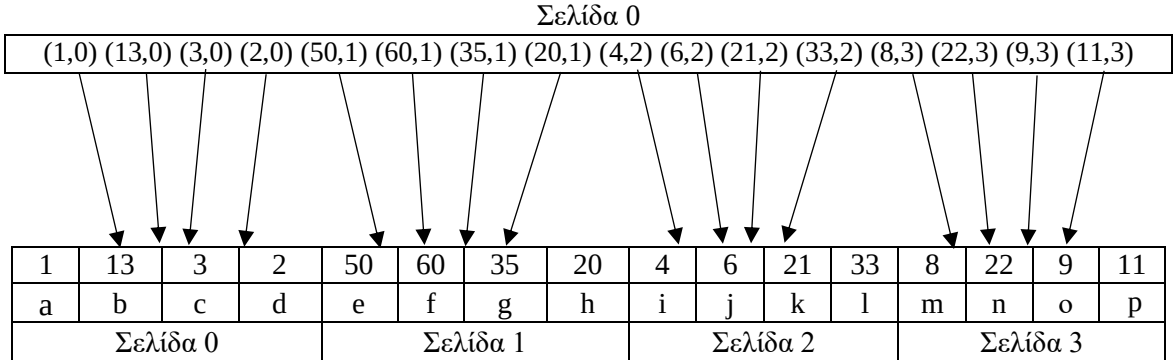
Η επεξεργασία του αρχείου δεν γίνεται απευθείας στον δίσκο αλλά μεταφέροντας στην κεντρική μνήμη κάθε φορά μία σελίδα αρχείου το οποίο σημαίνει μία πρόσβαση στο αρχείο (ένα disk access). Η σελίδα διαβάζεται σε έναν buffer μεγέθους 128 bytes που μετατρέπεται σε πεδίο 4 εγγραφών. Η επεξεργασία της σελίδας γίνεται πάνω στο buffer στην κεντρική μνήμη. Αν χρειαστεί να μεταφερθεί στο αρχείο, το buffer ξαναγράφεται ως σελίδα δίσκου πίσω στο αρχείο (ένα disk access). Η απόδοση μιας μεθόδου επεξεργασίας εξαρτάται από τον αριθμό προσβάσεων σε σελίδες δίσκου (disk accesses) που αφορούν είτε διάβασμα είτε γράψιμο σελίδων στον δίσκο.

Ένα αρχείο μπορούμε να το διαβάσουμε σειριακά, διαβάζοντας κάθε φορά μία σελίδα (την μία σελίδα μετά την άλλη) και μεταφέροντας την κάθε σελίδα στην κεντρική μνήμη. Κάθε σελίδα έχει έναν αριθμό θέσης **pos** που ισούται με τον αριθμό bytes από την αρχή του αρχείου. Στο παραπάνω σχήμα, η σελίδα 0 έχει pos=0x128, η σελίδα 1 έχει pos=1x128, η σελίδα 2 έχει pos=2x128 και η σελίδα 3 έχει pos=3x128). Η πρόσβαση σε μία σελίδα γίνεται με μια εντολή seek (για μετάβαση στην θέση της σελίδας pos). Αμέσως μετά, η εντολή read με όρισμα buffer θα διαβάσει την σελίδα στην θέση pos. Η εκτέλεση της εντολής read προωθεί τον δείκτη αρχείου στην επόμενη σελίδα. Η επόμενη εντολή read θα διαβάσει την επόμενη σελίδα και η διαδικασία μπορεί να επαναλαμβάνεται μέχρι το τέλος του αρχείου.

**Β τρόπος οργάνωσης αρχείου:** Ένας άλλος τρόπος οργάνωσης της ίδιας πληροφορίας στον δίσκο είναι με την χρήση 2 αρχείων. Το αρχείο έχει διασπαστεί σε δύο νέα αρχεία. Το πρώτο περιέχει κλειδιά και το δεύτερο περιέχει την πληροφορία των κλειδιών. Ειδικότερα, το πρώτο αρχείο περιέχει ζεύγη της μορφής (κλειδί, δείκτης). Κάθε ζεύγος έχει μέγεθος 4 + 4 bytes. Ο δείκτης είναι ο αριθμός σελίδας του αρχείου αρχείο που περιέχει την πληροφορία του κλειδιού. Από το αρχικό σχήμα θα προκύψουν

**1)** ένα αρχείο κλειδιών με 1 σελίδα, αφού όλα τα κλειδιά μαζί με τους δείκτες χωράνε σε μία σελίδα. Αυτό δεν θα ισχύει αν τα κλειδιά είναι πολύ περισσότερα. Για παράδειγμα, το κλειδί 21 παριστάνεται από το ζεύγος (21,2) δηλαδή η αντίστοιχη πληροφορία περιέχεται στη 2η σελίδα του δεύτερου αρχείου.

**2)** ένα αρχείο πληροφορίας με 4 σελίδες. Κάθε σελίδα χωράει την πληροφορία 4 κλειδιών και μαζί το κλειδί (δηλαδή το κλειδί επαναλαμβάνεται στο δεύτερο αρχείο)




Το κέρδος με τον δεύτερο τρόπο οργάνωσης είναι ότι η αναζήτηση για την πληροφορία που παριστάνεται από ένα κλειδί εξαρτάται μόνο από την αναζήτηση στο αρχείο των κλειδιών. Επειδή το αρχείο των κλειδιών είναι μικρότερο (από το αρχείο της οργάνωσης Α), χρειάζονται λιγότερες προσβάσεις δίσκου (disk accesses) κατά την αναζήτηση στοιχείων και άρα επιτυγχάνει καλύτερη απόδοση σε αναζητήσεις.

Για παράδειγμα, αν ζητείται η πληροφορία που αντιστοιχεί σε ένα κλειδί εργαζόμαστε ως εξής

**1)** Βρίσκουμε το κλειδί με αναζήτηση στο αρχείο των κλειδιών. Φέρνουμε την σελίδα στην κεντρική μνήμη (σε ένα buffer με την εντολή read). Στην κεντρική μνήμη, διασχίζουμε το buffer (το μεταχειριζόμαστε ως array), βρίσκουμε το ζεύγος του κλειδιού και από εκεί την τιμή που δηλώνει σε ποια σελίδα του δεύτερου αρχείου θα βρούμε την πληροφορία του κλειδιού.

**2)** Στο δεύτερο αρχείο, προχωράμε (με την εντολή seek) στην θέση που βρήκαμε στο προηγούμενο βήμα και διαβάζουμε την σελίδα (σε ένα buffer με την εντολή read). Στην κεντρική μνήμη διασχίζουμε το buffer μέχρι να βρούμε το κλειδί μαζί με την πληροφορία του κλειδιού.

**Εισαγωγή στοιχείων:** Δημιουργήστε ένα αρχείο με $\mathbf{N = 10^4}$ κλειδιά στο δίσκο με τιμές από 1 έως $10^4$. **Καμία τιμή κλειδιού δεν πρέπει να επαναλαμβάνεται.** Οι τιμές παράγονται από μία γεννήτρια τυχαίων αριθμών που παράγει τιμές στο διάστημά 1 έως $10^6$.

Στην διάρκεια της εισαγωγής στοιχείων για την δημιουργία των αρχείων προχωράμε σταδιακά. Η αποθήκευση της πληροφορίας γίνεται ανά μία σελίδα κάθε φορά. Κάθε νέα σελίδα γράφεται στο τέλος του αρχείου αμέσως μετά την τελευταία. Στην κεντρική μνήμη έχετε ένα buffer μεγέθους 128 bytes (όσο το μέγεθος της σελίδας) το οποίο κάθε φορά γεμίζει με την πληροφορία που θέλετε να γράψετε στο αρχείο. Μόλις γεμίσει το buffer, το περιεχόμενο του μεταφέρεται στο αρχείο με την εντολή write. Στην συνέχεια, αδειάζουμε το buffer, γράφουμε τα επόμενα δεδομένα και τα μεταφέρουμε ξανά στο αρχείο. Κατασκευάστε τα αρχεία με τον Α και Β τρόπο οργάνωσης. Υπολογίστε τον αριθμό σελίδων των αρχείων που θα δημιουργηθούν με τον Α και Β τρόπο οργάνωσης.

**Αναζήτηση τυχαίας τιμής κλειδιού στο αρχείο:** Παρακάτω, θα χρησιμοποιήσουμε τους διαφορετικούς τρόπους οργάνωσης Α και Β για να διαπιστώσουμε ποιος τρόπος είναι πιο αποδοτικός σε εφαρμογές που ζητείται η εύρεση τυχαίου κλειδιού. Πιο αποδοτικός τρόπος είναι αυτός που επιτυγχάνει μικρότερο αριθμό προσβάσεων στο δίσκο (disk accesses).

Κάθε ερώτηση εκφράζεται με μία **τυχαία τιμή κλειδιού στο διάστημα 1 έως $10^6$ από αυτές που ήδη υπάρχουν.** Επειδή η απόδοση μιας πράξης αναζήτησης δεν είναι πάντα χαρακτηριστική της απόδοσης, θα κάνουμε 20 αναζητήσεις σε 20 τιμές κλειδιών και θα μετρήσουμε το μέσο όρο του

αριθμού προσβάσεων. Για το σκοπό της άσκησης, θα χρησιμοποιήσετε 20 κλειδιά με τιμές που υπάρχουν ήδη στο αρχείο. Μετρήστε το μέσο όρο αριθμού προσβάσεων που απαιτούνται για την εύρεση κλειδιού εγγραφής στον Α και Β τρόπο οργάνωσης.

Για την σειριακή αναζήτηση, ανοίξτε το αρχείο και διαβάστε την πρώτη σελίδα του αρχείου στον buffer της κεντρικής μνήμης (αυτό κοστίζει μία πρόσβαση στο δίσκο). Ψάξτε στον buffer στην κεντρική μνήμη. Αν το κλειδί δεν υπάρχει στον buffer τότε επαναλαμβάνεται ή ίδια διαδικασία για την επόμενη σελίδα του δίσκου, και αν χρειαστεί ξανά μέχρι να εξαντληθεί το αρχείο. Μετρήστε τον αριθμό προσβάσεων που χρειάστηκαν μέχρι να βρεθεί το κλειδί. Μετρήστε τον μέσο αριθμό προσβάσεων για όλες τις αναζητήσεις. Ειδικότερα, μετρήστε τον μέσο αριθμό προσβάσεων που θα χρειαστείτε για τους τρόπους οργάνωσης Α και Β.

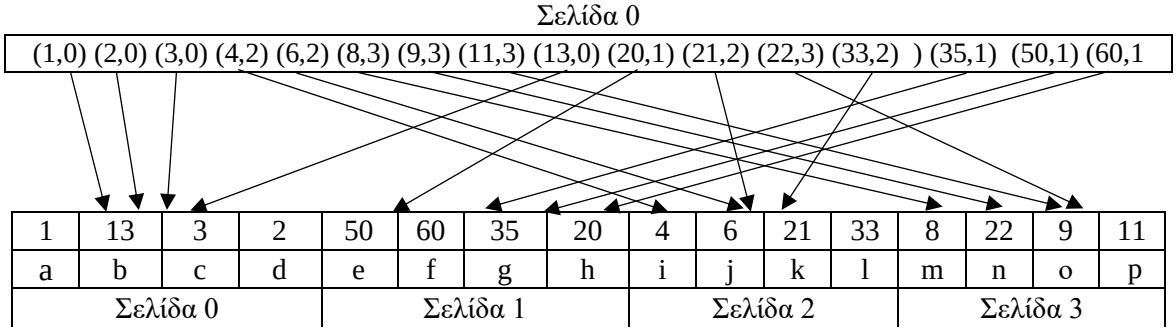
**Οργάνωση πληροφορίας σε ταξινομημένο αρχείο και απόδοση αναζήτησης (3 μονάδες)**

**Γ τρόπος οργάνωσης αρχείου:** Εφαρμόστε τον αλγόριθμο «εξωτερικής ταξινόμησης» (external sorting) που παρουσιάστηκε στο φροντιστήριο στον **Α τρόπο οργάνωσης**. Το κριτήριο ταξινόμησης εφαρμόζεται πάνω στις τιμές των κλειδιών (και όχι στην πληροφορία). Εναλλακτικά μπορείτε να βρείτε έναν αλγόριθμο ταξινόμησης αρχείων από το διαδίκτυο (δεν είναι απαραίτητο να τον φτιάξετε εσείς εκτός από το ότι θα πρέπει να προσαρμόσετε τον κώδικα να χειρίζεται εγγραφές με κλειδιά). Σε αυτή την περίπτωση, πρέπει να αναφέρετε ποιος είναι αυτός ο αλγόριθμος και σε ποια διεύθυνση υπάρχει. Θα χρειαστεί να μετρήσετε τον αριθμό των προσβάσεων δίσκου που χρειάστηκαν κατά την ταξινόμηση.

Θα εφαρμόσετε τον αλγόριθμο ταξινόμησης στον Α τρόπο οργάνωσης αρχείου. Το αποτέλεσμα στο παράδειγμα φαίνεται στο παρακάτω σχήμα. Το κριτήριο ταξινόμησης εφαρμόζεται πάνω στο κλειδί και προκύπτει ένα νέο αρχείο στο οποίο τα κλειδιά εμφανίζονται με αύξουσα σειρά.

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 6 | 8 | 9 | 11 | 13 | 20 | 21 | 33 | 35 | 35 | 50 | 60 |
| a | d | c | i | j | m | o | p | b | h | k | l | g | g | e | f |
| Σελίδα 0 | | | | Σελίδα 1 | | | | Σελίδα 2 | | | | Σελίδα 3 | | | |

**Δ Τρόπος οργάνωσης αρχείου:** Στην συνέχεια θα εφαρμόσετε τον αλγόριθμο ταξινόμησης στο Β τρόπο οργάνωσης αρχείου. Το αποτέλεσμα στο παράδειγμα φαίνεται στο παρακάτω σχήμα. Στην πράξη, **η ταξινόμηση εφαρμόζεται μόνο στο πρώτο αρχείο** (το αρχείο κλειδιών) ενώ το δεύτερο αρχείο μένει ως έχει. Προκύπτει ένα νέο αρχείο κλειδιών στο οποίο τα κλειδιά εμφανίζονται με αύξουσα σειρά. Το αρχείο με την πληροφορία δεν αλλάζει. Το αποτέλεσμα φαίνεται στο παρακάτω σχήμα.




| 1 | 13 | 3 | 2 | 50 | 60 | 35 | 20 | 4 | 6 | 21 | 33 | 8 | 22 | 9 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p |
| Σελίδα 0 | | | | Σελίδα 1 | | | | Σελίδα 2 | | | | Σελίδα 3 | | | |

**Δυαδική αναζήτηση στο αρχείο:** Για την δυαδική αναζήτηση, ανοίξτε το αρχείο και υπολογίστε τον αριθμό σελίδων του αρχείου των κλειδιών. Για κάθε αναζήτηση, εκμεταλλευτείτε ότι το αρχείο των κλειδιών είναι ταξινομημένο: διαβάστε αρχικά την μεσαία σελίδα του αρχείου στην κεντρική μνήμη. Εξετάστε αν το κλειδί είναι στην σελίδα (με δυαδική αναζήτηση). Αν υπάρχει, η αναζήτηση σταματάει. Αν το κλειδί που ψάχνετε είναι μικρότερο από τον μικρότερο αριθμό της σελίδας, τότε φέρτε στην κεντρική μνήμη την μεσαία σελίδα του αριστερού μισού του αρχείου. Αν το κλειδί είναι μεγαλύτερο από τον μεγαλύτερο αριθμό της σελίδας, φέρτε στην κεντρική μνήμη την μεσαία σελίδα του δεξιού μισού του αρχείου. Συνεχίστε με τον ίδιο τρόπο την αναζήτηση. Μετρήστε τον μέσο αριθμό προσβάσεων για όλες τις αναζητήσεις. Εφαρμόστε δυαδική αναζήτηση για εύρεση τυχαίου κλειδιού, Μετρήστε την μέση τιμή της απόδοσης για τον τρόπο οργάνωσης Γ και Δ.

**Παρατήρηση**: ο αλγόριθμος εξωτερικής ταξινόμησης μπορεί να επεξεργαστεί κάθε μέγεθος αρχείου. Στην άσκηση το αρχείο χωράει στην κεντρική μνήμη και μπορεί να διαβαστεί ολόκληρο σε ένα πεδίο (array) εγγραφών που η κάθε μία έχει μέγεθος 32 bytes. Ταξινομήστε το πεδίο στην κεντρική μνήμη (με όποιο αλγόριθμο θέλετε) και γράψτε το στον δίσκο.

### Τεκμηρίωση των Αποτελεσμάτων (2 μονάδες)

Συγκεντρώστε τα αποτελέσματα στον παρακάτω πίνακα και προσπαθήστε να δικαιολογήστε την απόδοση κάθε μεθόδου. Εξηγήστε για ποιο λόγο μία μέθοδος είναι πιο αποδοτική από μία άλλη για αναζήτηση (συγκρίνετε τις μεθόδους Α, Β, Γ και Δ του πίνακα). Εξηγήστε ποια μέθοδος ταξινόμησης είναι πιο αποδοτική (περιπτώσεις Γ και Δ). Συνολικά δεν χρειάζεται να γράψετε πάνω από 1 ή 2 σελίδες αρκεί κάθε απάντηση να είναι σαφής και αιτιολογημένη σωστά. Δεν χρειάζεται να γράψετε περισσότερα σχόλια για την υλοποίηση του κώδικα εκτός από ότι ζητείται παρακάτω (δείτε την ενότητα «παραδοτέα»).

| Μέθοδος | Α. τρόπος οργάνωσης αρχείου | Β. τρόπος οργάνωσης αρχείου | Γ. τρόπος οργάνωσης αρχείου | Δ. τρόπος οργάνωσης αρχείου | Απόδοση Εξωτερικής Ταξινόμησης περίπτωση Γ | Απόδοση Εξωτερικής Ταξινόμησης περίπτωση Δ |
|---|---|---|---|---|---|---|
| Απόδοση (αριθμός προσβάσεων δίσκου) | | | | | | |

**<u>Παραδοτέα:</u>** Ένα συμπιεσμένο zip αρχείο που περιέχει ότι ζητείται παρακάτω:

- Ο κώδικας περιέχει συνοπτικά σχόλια που εξηγούν την υλοποίηση.
- Μία έκθεση που περιγράφει σε 1-2 σελίδες πως φτιάχτηκε ο κώδικας (δηλ. για κάθε ερώτημα ποια είναι η γενική ιδέα της λύσης σε 3-4 προτάσεις), υπάρχουν σαφείς οδηγίες μετάφρασης από compiler και εκτέλεσης, τι λάθη έχει (αν έχει, περιπτώσεις που δεν δουλεύει το πρόγραμμα, ή περιπτώσεις που κάνει περισσότερα από όσα σας ζητεί η άσκηση, τι χρησιμοποιήσατε από έτοιμα προγράμματα ή πηγές πληροφόρησης. Υποδείξετε ακόμα και πηγές στο WWW όπως Wikipedia (πλήρεις διευθύνσεις) ή ακόμα και συναδέλφους που σας βοήθησαν στην άσκηση.
- Στην ενότητα τεκμηρίωσης των αποτελεσμάτων πρέπει να υπάρχει απόλυτη σαφήνεια
- Εκτός των παραπάνω, οι ασκήσεις βαθμολογούνται με άριστα εφόσον:
  - Το zip είναι πλήρες
  - Οι κώδικες περνούν από compiler και εκτελούνται σωστά σε windows ή Linux
  - Ο κώδικάς σας δουλεύει για οποιοδήποτε αρχείο δεδομένων που θα σας δοθεί ως είσοδος, και όχι μόνο το αρχείο που παράγετε εσείς